



خريف 2019

البرمجة الشيئية

Object Oriented Programming (with Java)

ITGS211

المحاضرة السابعة

الفصل الدراسي: خريف 2019

1

خصائص البرمجة الشيئية

- 1 . تمثل نظام محاكاة للعالم الحقيقي.
- 2 . نستطيع من خلالها إخفاء تفاصيل البرمجة والتي يطلق عليها التغليف (*Encapsulation*)
- 3 . نستطيع من خلالها إعادة استخدام الكود مرة أخرى عن طريق الوراثة (*Inheritance*)
- 4 . يمكن من خلالها استخدام اسم واحد يمثل عدة وظائف مختلفة (*Function Overloading , Polymorphism*)

2

Classes

الـ **class** هو أساس البرمجة الشيئية (OOP) والذي يُبنى عليه البرنامج.

• تعريف الـ Class يكون على الشكل:

```
AccessModifier class ClassName
{
    [fields declaration]        \\ الخصائص
    [methods declaration]       \\ العمليات
}
```

- عن طريق الـ Class يُمكن للمُبرمج استحداث عدد من المتغيرات مختلفة النوع تُسمى objects.

3

تصميم الـ Class

عند تصميم Class يجب تحديد ما يلي:

- البيانات (data) التي يجب أخذها بالاعتبار.
 - الاجراءات أو الأفعال (actions) التي يجب انجازها.
 - البيانات (data) التي يمكن تعديلها (modify).
 - البيانات (data) التي يجب أن يُسمح بالوصول إليها.
 - أي قواعد بشأن الكيفية التي ينبغي بها تعديل البيانات.
- عادةً يتم تصميم الـ Class بالاستعانة بمخطط لغة النمذجة الموحدة (UML).

4

الخصائص Attributes

- عناصر البيانات للـ class تُحدد الكائن ليكون instance من الـ class.
- الخصائص (attributes) يجب أن تكون ومُميزة للـ class ومُعروفة له.
- مثلاً: يتم تعريف المستطيل عن طريق:
 - الطول length.
 - العرض width.
- يتم الوصول إلى الخصائص (attributes) عن طريق الدوال methods داخل الـ class.
- كقراءة الطول والعرض ثم حساب محيط ومساحة المستطيل.

5

الدوال Methods

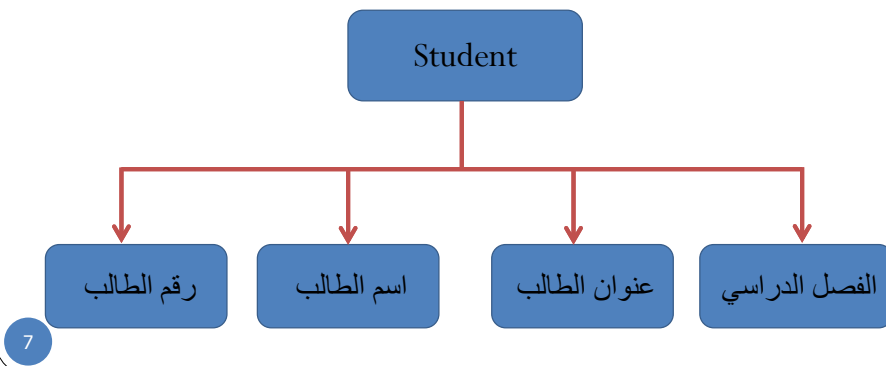
- الدوال تحقق العمليات التي يمكن أن تتم على الخصائص (البيانات) داخل الـ Class.
- الدوال التي يُمكن استخدامها من قِبل classes أخرى يتم جعلها عامة public.
- شكل رأس الدالة (Method header)

```
AccessModifier Return Type MethodName(Parameters)
{
    // Method body.
}
```

6

مثال Student Class

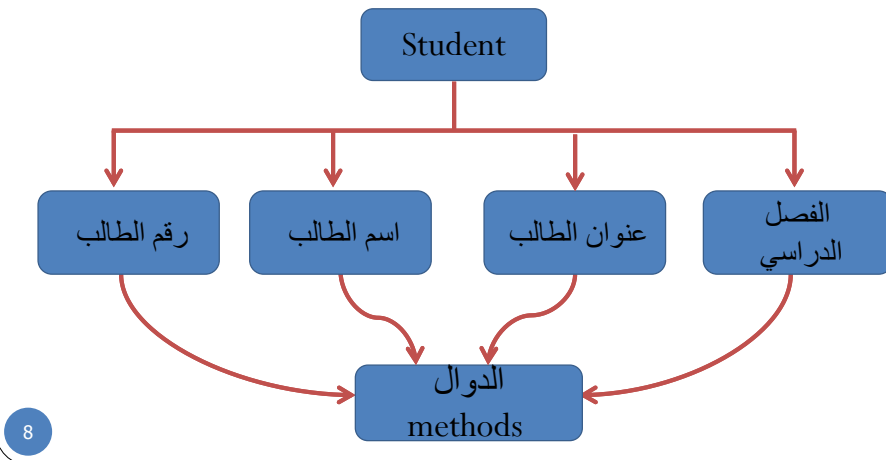
في الـ Student Class بياناته سوف تكون بيانات الطالب العامة
مثل رقم الطالب، اسم الطالب، عنوان الطالب، الفصل
الدراسي،... الخ



7


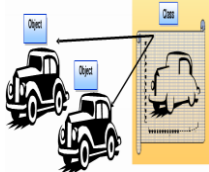
مثال Student Class

في الـ Student Class الدوال التي تحقق العمليات التي تتم على
الطالب داخل مثل عملية إضافة أو حذف أو تعديل بيانات.



8

مثال Car class

Type="BMW"

Model=2015

Price=10000

MilesDrive=105

Owner="Hussein"

GetPrice()

GetOwner()

```

public class Car {
    String Type;
    int Model;
    double Price;
    double MilesDrive;
    String Owners;
    double GetPrice(){
        double NewPrice=Price-(MilesDrive*100);
        return NewPrice;}
    String GetOwner(){
        return owners;}
}

```

```

Car car2= new Car();
car2.Type="BMW";
car2.Model=2015;
car2.Price=10000;
car2.MilesDrive=105;
car2.Owner="Hussein";

```

مثال Circle class

```

public class Circle {
    // بيانات الدائرة
    public double x, y; // مركز الدائرة
    public double r; // نصف قطر radius الدائرة } fields declaration

    // دوال (Methods) لارجاع محيط ومساحة الدائرة
    public double circumference() {
        return 2*3.14*r; }
    public double area() {
        return 3.14 * r * r; }
}

```

Method Body

} methods declaration

10

في هذا الـ class تم الإعلان عن متغيرات x,y,z وتم انشاء دالة بالاسم Printxyz وفيها تتم طباعة المتغيرات x,y,z.

```
public class class1 {
    int x, y,z;
    void Printxyz()
    {
        System.out.println("x= " + x);
        System.out.println("y= " + y);
        System.out.println("z= " + z);
    }
}
```

} fields declaration

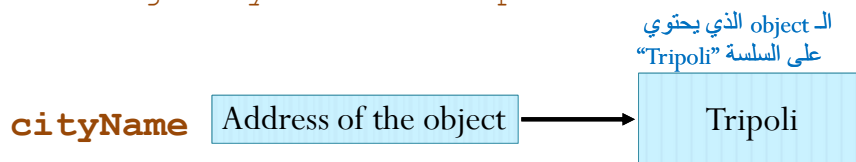
} methods declaration

11

Classes and Instances

كما سبق وأن تعلمنا أن الـ reference variable يحتوي عنوان الـ object .

```
String cityName = "Tripoli"
```



الدالة length () للـ String class تُرجع قيمة صحيحة تساوي طول السلسلة. فمثلاً:

```
int stringLength = cityName.length();
```

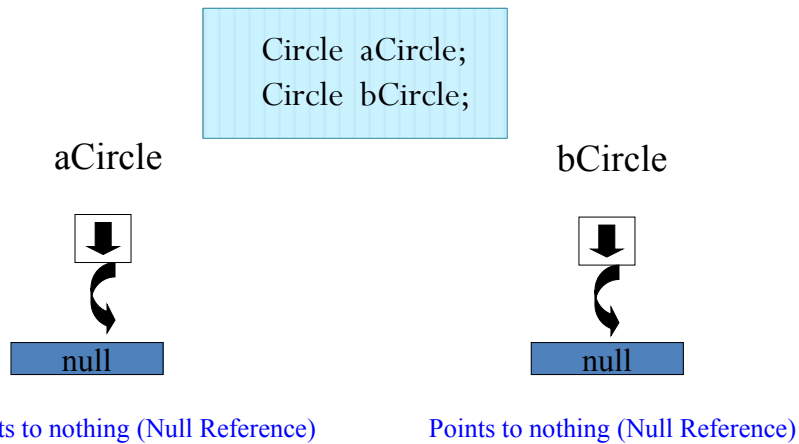
المتغير stringLength سيحمل القيمة 7، حيث أن السلسلة "Tripoli" تحوي 7 أحرف. كما يلي:

T	r	i	p	o	l	i
1	2	3	4	5	6	7

12

عودة إلى مثال Circle class (الشرح 9)

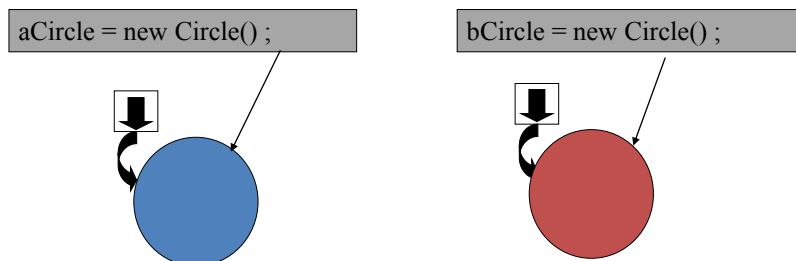
- `aCircle`, `bCircle` تشير إلى كائن `Circle` و ليس الكائن نفسه.



13

انشاء objects من ال-class

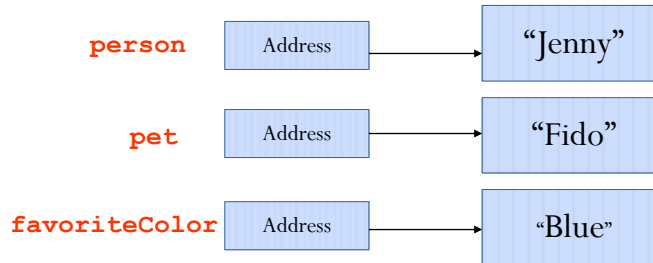
- الكائنات أنشأت باستخدام الكلمة المحجوزة `new`.
- الكائنات `aCircle` و `bCircle` تشير إلى كائنات `Circle`.



14

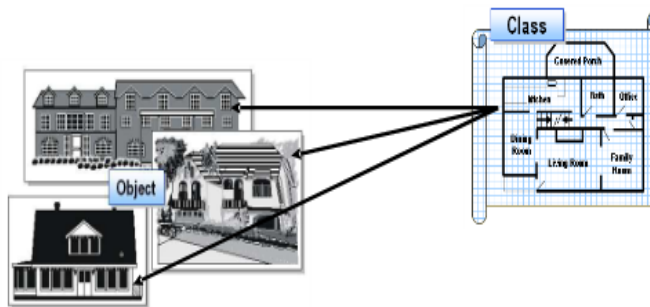
- إذا عدة objects يمكن انشائها من class. و كل object مستقل عن غيره (independent). على سبيل المثال:

```
String person = "Jenny";
String pet = "Fido";
String favoriteColor = "Blue";
```



15

- كل instance من class يحتوي على بيانات مختلفة.
- ال instances كلها مشتركة في نفس التصميم (design).
- ولكل instance كل الصفات (attributes) والدوال (methods) التي تم تعريفها في ال class.
- يتم تعريف Classes لتمثيل مفهوم (concept) واحد.



16

كيفية استعمال الـ class1

```
package oop1;
public class Oop1 {
    public static void main(String[] args) {
        class1 x = new class1();
        x.a=20;
        x.b=40;
        x.c=30;
        x.printabc();
    }
}

class class1{
    int a,b,c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

17

في هذا المثال:

-بعد إنشاء الكلاس class1 وتحديد متغيراته ودواله.

-تم استعمال الـ class1 داخل الدالة الرئيسية وتم الإعلان عن متغير باسم x من نوع class1 وفي هذه الحالة لا يسمى متغير بل يسمى كائن object مع استعمال كلمة new لإنشاء الكائن x.

a=20
b=40
c=30

مثال: اكتب برنامج العمليات الحسابية الأربع باستخدام الـ class

```
class mathoperation {
    double result;
    double sum( double v1,double v2) {
        result =v1+v2;
        return result;
    }
    double subtract ( double v1,double v2) {
        result =v1-v2;
        return result;
    }
    double multiple( double v1,double v2) {
        result =v1*v2;
        return result;
    }
    double divisor ( double v1,double v2) {
        result =v1/v2;
        return result ;
    }
}
```

18

← (1) إنشاء الـ class

(2) البرنامج الرئيسي وتعريف الكائن mathobject من mathoperation

```

package oop2
import java.util.Scanner;
public class oop2 {
    public static void main(String[] args) {
        mathoperation mathobject =new mathoperation ();
        Scanner input=new Scanner(System.in);
        double N1=input.nextDouble();
        double N2=input.nextDouble();

        System.out.println("\n Sum is :"+ mathobject.sum(N1,N2));
        System.out.println("\n subtract is :"+ mathobject.subtract(N1,N2));
        System.out.println("\n multiple is :"+ mathobject.multiple(N1,N2));
        System.out.println("\n divisor is :"+ mathobject.divisor(N1,N2));
    }
}

```

19

مثال: اكتب برنامج يعمل على طباعة مضروب عدد يتم ادخاله قبل من المستخدم.

```

import java.util.Scanner;
public class oop3 {
    public static void main(String[] args) {
        Fact fact =new Fact ();
        Scanner input=new Scanner(System.in);
        int N1=input.nextInt();
        System.out.println("Fact is :"+ fact.f2(N1));
    }
}

class Fact {
    static int f2(int x) {
        int f=1,i;
        for (i=1;i<=x;i++)
            f=f*i;
        return f ;
    }
}

```



```

run:
5
Fact is :120
BUILD SUCCESSFUL

```

20

دالة البناء Construction

وهي دالة في class، تحمل نفس اسم class ولا تُرجع قيمة، تنفذ تلقائياً عند استعمال class، أي تُستدعى عند إنشاء objects وتجدها بعد كلمة new وظيفتها تسجيل أي قيم ابتدائية أو تعريف أي متغيرات أو أي شروط ابتدائية. إذا لم يتم المبرمج بكتابتها ضمن class فسيتم استدعاء دالة بناء Constructor افتراضية.

فمثلاً: عندما نقوم بإنشاء كائن Student من class1 بواسطة الكلمة new فإن دالة البناء constructor يتم استدعائه مباشرة.

class1 student=new class1();

دالة البناء

21

مثال: استعمال class Student للإعلان عن كائنين (Ahmed and Khalid) لحساب متوسط الدرجات لكل كائن في مادتين ثم طباعة رقم الطالب والمعدل داخل البرنامج؟

```
class student {
    int id, oop, math;
    int average() {
        int x=(oop+ math)/ 2;
        return x;
    }
}
```

(1) إنشاء الكلاس student ويتكون من ثلاث متغيرات ودالة.

```
package oop1;
public class Oop1 {
    public static void main(String[] args) {
        Student ahmed =new Student();
        Student khalid =new Student();
    }
}
```

(2) إنشاء object من class

دالة البناء

لاحظ عدم ذكر دالة البناء student() عند إنشاء class المُسمى student ومع ذلك تم استدعائها هنا عند إنشاء الكائن ahmed وكذلك الكائن khalid.

22

(3) استخدام الكائن داخل البرنامج

```

ahmed .id =0234567;
ahmed .oop =80;
ahmed .math =60;
System .out.println("ahmed id =" + ahmed.id +" average="+ ahmed.average());

khalid .id =4536871;
khalid .oop =70;
khalid .math =80;
System .out.println("khalid id =" + khalid.id +" average="+ khalid.average());

```

23

البرنامج كامل للمثال السابق

```

package oop_example;
public class OOP_Example {
    public static void main(String[] args) {
        Student ahmed =new Student();
        Student khalid =new Student();
        ahmed .id =0234567;
        ahmed .oop =80;
        ahmed .math =60;
        System.out.println("ahmed id =" + ahmed.id +" ,average="+ ahmed.average());
        khalid.id =4536871;
        khalid.oop =70;
        khalid.math =80;
        System.out.println("khalid id =" + khalid.id +" , average="+ khalid.average());
    }
}

class Student {
    int id,oop,math;
    int average(){
        int x=(oop+ math)/ 2;
        return x;
    }
}

```

Output - OOP_Example (run)

```

run:
ahmed id =80247 ,average=70
khalid id =4536871 , average=75
BUILD SUCCESSFUL (total time: 0 seconds)

```

24

- إذاً **دالة البناء** هي من الخصائص التي تدعمها البرمجة الشيئية داخل الـ class وهي دالة كأى دالة **ولكن** تنفذ تلقائياً دون استدعاء أي بمجرد تعريف كائن من الـ class. **ويجب** أن يكون لها نفس اسم الـ class الذي تقع ضمنه، وبما أنها دالة **فقد يكون لها مدخلات parameters أو لا يكون.**

Class Myconstructor

```
{
    Myconstructor() {
        System.out.println("لقد تم تنفيذ هذه العملية باستخدام دالة البناء");
    }
}
```

25

استخدام دالة البناء

```
public class Constructor {
    public static void main (String [ ] args)
    {
        Myconstructor myconstructor = new Myconstructor();
    }
}
class Myconstructor
{
    Myconstructor()
    {
        System.out.println ("لقد تم تنفيذ هذه العملية باستخدام دالة البناء");
    }
}
```

26

The screenshot shows an IDE window titled 'Constructor.java'. The source code is as follows:

```

1 public class Constructor {
2     public static void main (String [ ] args)
3     {
4         Myconstructor myconstructor = new Myconstructor();
5     }
6 }
7 class Myconstructor
8 {
9     Myconstructor()
10    {
11        System.out.println ("لقد تم تنفيذ هذه العملية باستخدام دالة البناء");
12    }
13 }
14

```

The output window at the bottom shows the following text:

```

run:
لقد تم تنفيذ هذه العملية باستخدام دالة البناء
BUILD SUCCESSFUL (total time: 0 seconds)

```

```

public class testclass {

    public testclass() {
        System.out.println("I'm constructor!");
    }

}

public class testoop {

    public static void main(String[] args) {

        // TODO, add your application code
        System.out.println("Hello World!");
        testclass obj1 = new testclass();

    }

}

```

هذا الـ **testclass** يقوم فقط بإظهار رسالة تعريفية عنه

هذا الـ **Class** الرئيسي ومنه يتم استدعاء الـ **testclass** وهذا فقط يتم التفاعل والاستفادة من **testclass**

General Output

```

-----Configuration: testoop.
Hello World!
I'm constructor!

Process completed.

```

الناتج وتلاحظ فيه الرسالة المكتوبة في **testclass**

28

استخدام دالة البناء بدون مُدخلات (بارامترات)

```
public class Constructor1 {
    public static void main (String [ ] args){
        Myrectangle rect = new Myrectangle();
        System.out.println ("width="+ rect.width+"\t height="+ rect.height);
    }
}

class Myrectangle {
    double width,height;
    Myrectangle(){
        width=10;
        height=2 ;
    }
}
```

إنشاء object من الـ class وفيه استدعاء دالة البناء

إنشاء دالة البناء داخل الـ class ولها نفس اسمه وليس لها مُدخلات.

29

استخدام دالة البناء ولها مدخلات

```
public class Constructor1 {
    public static void main (String [ ] args)
    {
        Myrectangle rect = new Myrectangle(20,4);
        System.out.println ("width="+ rect.width+"\t height="+ rect.height);
    }
}

class Myrectangle
{ double width,height;
    Myrectangle(double w , double h )
    {
        width=w;
        height=h ;
    }
}
```

إنشاء object من الـ class وفيه استدعاء دالة البناء ولها مدخلات

إنشاء دالة البناء داخل الـ class ولها نفس اسمه ولها مُدخلات.

Output:
width=20.0 height =4.0

30

تعريف كائن إضافي وهو rect2

```
public class Constructor1 {
    public static void main (String [ ] args)
    {
        Myrectangle rect = new Myrectangle(20,4);
        Myrectangle rect2 = new Myrectangle(40,8);
        System.out.println ("width="+ rect.width+"\t height="+ rect.height);
        System.out.println ("width2="+ rect2.width+"\t height2="+ rect2.height); }
    }
    class Myrectangle
    { double width,height;
        Myrectangle(double w , double h )
        {
            width=w;
            height=h ;
        }
    }
}
```

Output:

```
width=20.0      height =4.0
width2=40.0     height2 =8.0
```

31

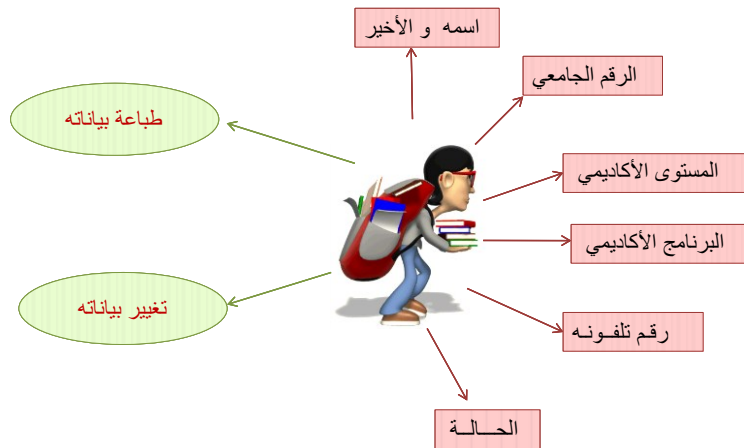
بناء و استخدام الـ Classes

انشيء نوع جديد باسم Student تتمكن من خلاله من تمثيل أي طالب بالكلية مع توفير الدوال التي يمكننا من التعامل مع بياناته.



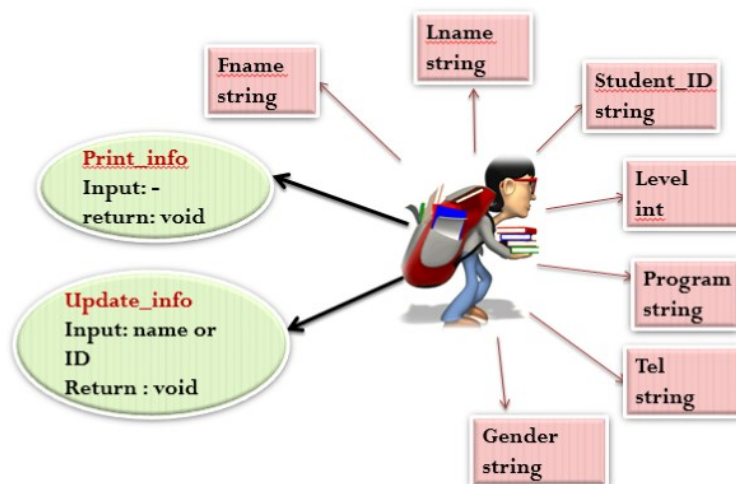
32

التفكير



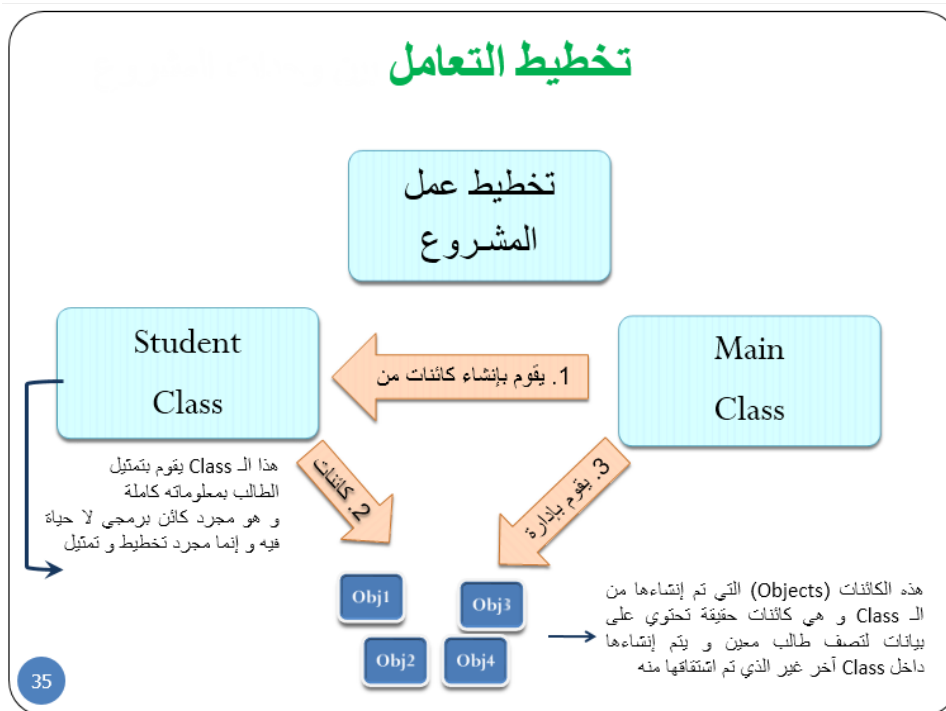
33

التخطيط



34

تخطيط التعامل



التنفيذ

بناء Student Class

```
public class student {
    private String fname;
    private String lname;
    private String tel;
    private String stu_id;
    private String Program;
    private String Gender;
    private int level;

    public student() {
        fname = "First Name";
        lname = "Last Name";
        tel = "000000000";
        stu_id = "120090000";
        Program = "Deploma";
        Gender = "male";
        level = 1;
    }

    public student(String fn, String ln, String tele, String id, String prog, String gen, int lev) {
        fname = fn;
        lname = ln;
        tel = tele;
        stu_id = id;
        Program = prog;
        Gender = gen;
        level = lev;
    }
}
```

Instance variable

Constructor

Constructor

هذه المتغيرات المحلية الخاصة بالـ Class و التي تمثل صفات هذا الـ Class و يكون لها حدود للمشاهدة من خلال استخدام Access Modifiers

هذه دالة بدء الـ Class أي دالة البناء والتي من خلالها يتم إعطاء قيم ابتدائية للكائنات (Objects) التي يتم إنشاؤها من هذا الـ Class، و مثلها مثل كافة الدوال يمكن أن يتم إنتاج صور متعددة منها كما تشاهد.

36

التنفيذ

```

public void print ()
{
    System.out.println("The information is \n" +
        "Name: "+fname+" "+lname+"\n"+
        "ID: "+stu_id+"\n"+
        "Program: "+Program+"\n"+
        "Level: "+level+"\n"+
        "Gender: "+Gender+"\n"+
        "Tel.: "+tel);
}

public void update( String first_name, String Last_name )
{
    fname = first_name;
    lname = Last_name;
    System.out.println("The name is update ");
}

public void update( String ID )
{
    stu_id = ID;
    System.out.println("The ID is update ");
}

```

Method

Overloaded Method

هذه دالة لطباعة بيانات الكائنات التي سيتم إنشاؤها من هذا الـ Class

هذه دالة لتعديل اسم أو رقم الكائنات التي سيتم إنشاؤها من هذا الـ Class علماً بأن لها صور متعددة لاختلاف طريقة التعديل

37

التنفيذ

```

public class studentProj {
    public static void main(String[] args) {
        student st1 = new student();
        st1.print();
        student st2 = new student("Mahmoud", "Alfarra", "0599000000", "T1000200", "Master", "Male", 2);
        st2.print();
        st1.update("Ali", "Hassan");
    }
}

```

بهذه الجملة يتم إنشاء كائن جديد من Student اسمه st1 معتمداً على القيم الافتراضية الموجودة داخل Student بعد ذلك يتم التعامل مع st1

بهذه الجملة يتم إنشاء كائن جديد من Student اسمه st2 مع إعطائه قيم من خلال استخدام الـ Overloaded constructor بعد ذلك يتم التعامل مع st2 كتعديل البيانات أو طباعتها

38

التغليف Encapsulation

- يجب أن يُحدد النطاق scope أو مدى رؤية الـ Classes حسب الحاجة.
- كل class يجب أن يحتوى على كل ما يحتاجه لعمله.
- ضم الخصائص (attributes) الدوال (methods) المناسبة داخل class واحد، يُسمى التغليف (encapsulation).
- التغليف يضمن أن يكون الـ class مكتفي ذاتياً (self-contained).

إخفاء البيانات (Data Hiding)

- هو جانب آخر من التغليف: فيه يجب أن لا يكون الـ class مكتفي ذاتياً (self-contained) **فقط ولكن** ينبغي أن يكون متمتع بتحكم ذاتي (self-governing) وبشكل جيد.
- الـ Classes تستخدم *private access* على الحقول لتخفيها عن باقي الـ Classes.
- للسماح بالوصول لبيانات الـ Class وتعديلها يحتاج إلى الدوال (methods).

39

البيانات القديمة Stale Data

- بعض البيانات تكون ناتجة من حسابات تجرى على عوامل مختلفة ، فمثلا مساحة (area) المستطيل ما هي إلا ناتج:
الطول (length) × العرض (width)
كلما تغيرت قيمة الطول أو العرض أو كليهما تغيرت قيمة المساحة ، ولأصبحت قيمة area قديمة ولا معنى لها لذلك أي خاصية قد تتغير بتغير عوامل أخرى وجب حسابها داخل دالة في الـ class لتجنب البيانات القديمة

```
public double getArea()
{
    return length * width;
}
```

40

التغليف Encapsulation

```
public class car {
    private int width=20;
    public int getdata( )
    {
        return width;
    }
    public void setdata (int w)
    {
        width=w;
    }
}
```

41

التغليف Encapsulation

```
Package student;
public class student {
    public static void main (String[ ] args)
    {
        car o1=new car ();
        o1.getdata();
        o1.setdata(40);
        System.out.println(o1.getdata());
    }
}
```

~~20~~

40

مفهوم **Encapsulation** هو الوصول إلى متغيرات من نوع **private** بطريقة غير مباشرة وذلك من خلال دوال في داخل الـ **class**.

42

التغليف Encapsulation

main - Navigator

- GetSet_Java
- main(String[] args)
- car
 - getdata() : int
 - setdata(int w)
 - doors : int
 - width : int

```
5  ...
6  ...
7  ...
9  ...
10 ...
11 ...
12 ...
13 class car {
14     ...
15     ...
16     ...
17     ...
18     ...
19     ...
20     ...
    ...
}
```

Output - GetSet_Java (run)

run:

```
3  ...
4  ...
5  ...
6  ...
7  ...
8  ...
9  ...
10 ...
11 ...
12 class car {
13     private int width=20;
14     int doors;
15     public int getdata() {
16         return width;
17     }
18     public void setdata (int w) {
19         width=w;
20     }
}
```

Output - GetSet_Java

run:

40

4

BUILD SUCCE

width has private access in car
(Alt-Enter shows hints)

```

1  public class GetSet_Java {
2      public static void main(String[] args) {
3          car o1=new car ();
4          o1.doors=4;
5          System.out.println(o1.doors);
6          o1.getdata();
7          o1.setdata(220);
8          System.out.println(o1.getdata());
9      }
10 }
11
12 class car {
13     private int width=20;
14     int doors;
15     public int getdata() {
16         return width;
17     }
18     public void setdata (int w){
19         width=w;
20     }
21 }

```

مفهوم التغليف
Encapsulation هو
الوصول إلى متغيرات من
نوع **private** بطريقة
غير مباشرة وذلك من خلال
دوال في داخل الـ **class**.

45

```

2  public class GetSet_Java {
3      public static void main(String[] args) {
4          car o1=new car ();
5          o1.doors=4;
6          System.out.println(o1.doors);
7          o1.getdata();
8          o1.setdata(220);
9          System.out.println(o1.getdata());
10     }
11 }
12 class car {
13     private int width=20;
14     int doors;
15     public int getdata() {
16         return width;
17     }
18     public void setdata (int w){
19         width=w;
20     }
21 }

```

Output - GetSet_Java (run) 88

run:
4
220
BUILD SUCCESSFUL (1)

46

الـClasses والمتغيرات (Variables) مجال الرؤية Scope

فيما يلي سرد لأنواع مجال الرؤية (Scope) للمتغير بناءً على مكان الاعلان عنه (declaration):

- داخل الدالة (Inside a method):
 - مرئي فقط ضمن الدالة method.
 - يُسمى متغير محلي (local variable).
- في مُعامل الدالة (In a method parameter):
 - يُسمى parameter variable
 - مثل المتغير المحلي (local variable) ومرئي فقط ضمن الدالة (method)
- داخل الـclass ولكن ليس داخل الدالة (Inside the class but not in a method):
 - مرئي لكل عناصر الـClass.
 - يُسمى instance field

47

Static Variable

- ✓ عند تعريف متغير ما في (Class) بـ static فهذا يعني أن جميع الكائنات (objects) التي سيتم اشتقاقها من هذا الصنف سيكون لهم قيمة مشتركة لهذا المتغير.
- ✓ و هذا على عكس ما يحدث في الوضع الطبيعي حيث يحصل كل كائن على نسخة افتراضية دون أن تتأثر بالكائنات الأخرى.

```
private static int count = 0;
```

48

Static Variable

```
class myclass{
    int a;
    int b;
    static int c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

class

a ☐
b ☐
c ☒

objects

a ☐
b ☐

a ☐
b ☐

a ☐
b ☐

49

Static Variable

```
class class1{
    int a,b;
    static int c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

عند تعريف متغير انه Static فهذا يعني انه مشترك بين كل الكائنات objects التي تنشأ من هذا Class، وأي تغير يطرأ عليه يظهر للجميع.

عند استخدامك هذا المتغير مع أي كائن object، سيظهر تلميح يوضح أنك تتعامل مع متغير Static

2

Accessing static field c

3

4

(Alt-Enter shows hints)

5

c void main(String[] args) {

6

new class1();

7

x.c=30;

8

x.a=20;

9

x.b=40;

لذلك من الأصح استخدام هذا المتغير مع اسم class المعرفة به بدلاً من اسم الكائن object، للدلالة على انه مشترك.

```
public static void main(String[] args) {
    class1 x = new class1();
    class1.c=30;
    x.a=20;
}
```

50

Static Variable

```
class class1{
    int a,b;
    static int c;
    void printabc()
    {
        System.out.println("a="+a);
        System.out.println("b="+b);
        System.out.println("c="+c);
    }
}
```

عند تعريف متغير انه Static فهذا يعني انه مشترك بين كل الكائنات objects التي تنشأ من هذا Class، وأي تغيير يطرأ عليه يظهر للجميع.

لاحظ النتائج تجد أن قيمة c في الحالتين ظهرت 500 رغم انها مع الكائن x كانت 30 ومع الكائن m كانت 500.

```
3 public static void main(String[] args) {
4     class1 x = new class1();
5     class1 M = new class1();
6     x.c=30;
7     x.a=20;
8     x.b=40;
9     M.a=200;
10    M.b=400;
11    M.c=500;
12    x.printabc();
13    M.printabc();
14 }
15 }
```

Output - OOp (run)

```
run:
a=20
b=40
c=500
a=200
b=400
c=500
BUILD SUCCESSFUL
```

51

Static Variable

```
public class StaticVariable {
    static int noOfInstances;
    StaticVariable() {
        noOfInstances++;
    }
}
```

```
public static void main(String[] args) {
    StaticVariable sv1 = new StaticVariable();
    System.out.println("No. of instances for sv1 : " + sv1.noOfInstances); // 1
    StaticVariable sv2 = new StaticVariable();
    System.out.println("No. of instances for sv1 : " + sv1.noOfInstances); // 2
    System.out.println("No. of instances for st2 : " + sv2.noOfInstances); // 2
    StaticVariable sv3 = new StaticVariable();
    System.out.println("No. of instances for sv1 : " + sv1.noOfInstances); // 3
    System.out.println("No. of instances for sv2 : " + sv2.noOfInstances); // 3
    System.out.println("No. of instances for sv3 : " + sv3.noOfInstances); // 3
}
```

52

```

4 public static void main(String[] args) {
5     StaticVariable sv1 = new StaticVariable();
6     System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
7     StaticVariable sv2 = new StaticVariable();
8     System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
9     System.out.println("No. of instances for sv2 : " + sv2.noOfInstances);
10    StaticVariable sv3 = new StaticVariable();
11    System.out.println("No. of instances for sv3 : " + sv3.noOfInstances);
12    System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
13    System.out.println("No. of instances for sv2 : " + sv2.noOfInstances);
14 }
15
16 class StaticVariable{
17     int noOfInstances;
18     StaticVariable()
19     {
20         noOfInstances++;
21     }
22 }

```

Output - StaticV (run)

```

run:
No. of instances for sv1 : 1
No. of instances for sv1 : 1
No. of instances for sv2 : 1
No. of instances for sv3 : 1
No. of instances for sv1 : 1

```

53

```

5 public static void main(String[] args) {
6     StaticVariable sv1 = new StaticVariable();
7     System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
8     StaticVariable sv2 = new StaticVariable();
9     System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
10    System.out.println("No. of instances for sv2 : " + sv2.noOfInstances);
11    StaticVariable sv3 = new StaticVariable();
12    System.out.println("No. of instances for sv3 : " + sv3.noOfInstances);
13    System.out.println("No. of instances for sv1 : " + sv1.noOfInstances);
14    System.out.println("No. of instances for sv2 : " + sv2.noOfInstances);
15 }
16
17 class StaticVariable{
18     static int noOfInstances;
19     StaticVariable()
20     {
21         noOfInstances++;
22     }
23 }

```

نفس المثال مع اضافة static قبل المتغير

Output - StaticV (run)

```

run:
No. of instances for sv1 : 1
No. of instances for sv1 : 2
No. of instances for sv2 : 2
No. of instances for sv3 : 3
No. of instances for sv1 : 3
No. of instances for sv2 : 3
BUILD SUCCESSFUL (total time: 0 seconds)

```

54

UML Class Diagram Unified Modelling Language

- الـ UML class diagram: هو أداة رسومية تُساعد في تصميم class.
- ويتكون المخطط من أقسام رئيسية كالتالي.

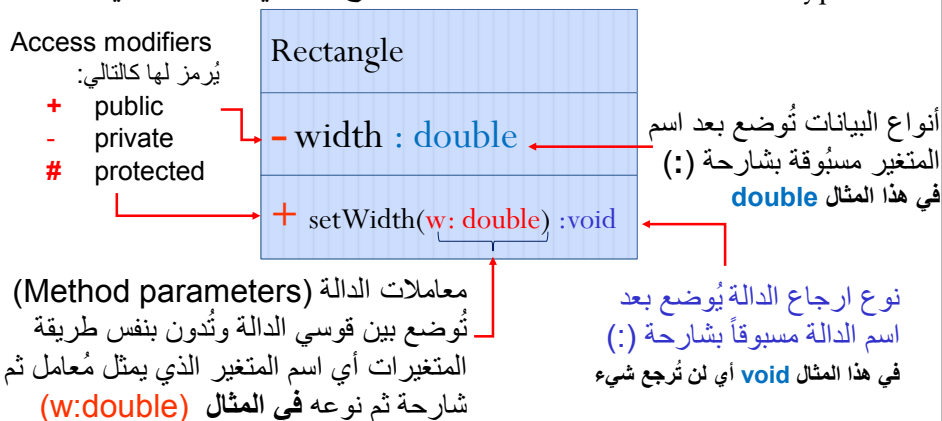
Class Name
Attributes
Methods

- من السهل جداً تحويل مخططات UML إلى Java Code.

55

UML Data Type and Parameter Notation

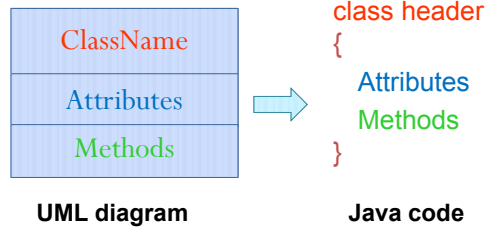
- UML diagrams هي لغة مستقلة، تستخدم تمثيل خاص لتوضيح return types و access modifiers.... إلخ كما في المثال التالي:



56

تحويل UML diagram إلى Code

وضع كل هذه المعلومات معاً، يُمكن من بناء Java class file وبكل سهولة باستخدام مُخطط UML. فأجزاء هذا المُخطط هي فعلاً مطابقة لهيكلية class file كما هو موضح أدناه.



57

بدايةً يمكن تجميع هيكلية الـ `class` واختبارها دون وجود `body` الدالة. فقط تأكد من وضع قيم وهمية للمرجعات الدالة التي لها نوع ارجاع غير الفراغ.

Rectangle
- width : double - length : double
+ setWidth(w : double) : void + setLength(len : double): void + getWidth() : double + getLength() : double + getArea() : double

```
public class Rectangle
{
    private double width;
    private double length;

    public void setWidth(double w)
    {
    }
    public void setLength(double len)
    {
    }
    public double getWidth()
    {
        return 0.0;
    }
    public double getLength()
    {
        return 0.0;
    }
    public double getArea()
    {
        return 0.0;
    }
}
```

58

Converting the UML Diagram to Code

Once the class structure has been tested, the method bodies can be written and tested.

Rectangle
- width : double - length : double
+ setWidth(w : double) : void + setLength(len : double): void + getWidth() : double + getLength() : double + getArea() : double

```
public class Rectangle
{
    private double width;
    private double length;

    public void setWidth(double w)
    {
        width = w;
    }
    public void setLength(double len)
    {
        length = len;
    }
    public double getWidth()
    {
        return width;
    }
    public double getLength()
    {
        return length;
    }
    public double getArea()
    {
        return length * width;
    }
}
```

إذا ما تم اختبار هيكلية الـ **Class**، يمكن كتابة آلية عمل كل دالة ومُرجعاتها واختبارها. 59

شكراً لإنصاتكم

أي سؤال؟
فالفائدة تبدأ بالسؤال ثم النقاش ..
لنبدأ النقاش فنستفيد ...

